Advantage360

KINESIS

DIRECT PROGRAMMING GUIDE

SmartSet Programming Engine

KB360

Proudly designed and hand-assembled in the USA since 1992

Kinesis® Advantage360™ Keyboard with the SmartSet™ Programming Engine

Keyboard models covered by this manual include all KB360 series keyboards (KB360-xxx). Some features may require a firmware upgrade. Not all features supported on all models. This manual does not cover setup and features for the Advantage360 Professional keyboard which features the ZMK programming engine.

August 26, 2022 Edition

This manual covers features included up through the firmware version 1.0.9.

If you have an earlier version of firmware, not all features described in this manual may be supported. To download the latest firmware here:

kinesis.com/support/adv360/#firmware-updates

KINESIS CORPORATION

22030 20th Avenue SE, Suite 102

Bothell, Washington 98021 USA

www.kinesis.com

FCC Radio Frequency Interference Statement

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a residential installation. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna
- Increase the separation between the equipment and receiver
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected
- Consult the dealer or an experienced radio/TV technician for help

Warning

To assure continued FCC compliance, the user must use only shielded interfacing cables when connecting to computer or peripheral. Also, any unauthorized changes or modifications to this equipment would void the user's authority to operate.

INDUSTRY CANADA COMPLIANCE STATEMENT

This Class B digital apparatus meets all requirements of the Canadian Interface-causing Equipment Regulations.

Cet Appareil numerique de la classe B respecte toutes les exiginces du Reglement sur le material broilleur du Canada.

1.0 Introduction

The Advantage360 is a fully-programmable keyboard that features onboard flash storage (the “v-Drive) and does not use any special drivers or software. The keyboard was designed to be programmed quickly and easily using the onboard shortcuts or via the SmartSet App for Windows and Mac. Power users have the option to bypass the SmartSet GUI and “Direct Program” the keyboard on all major operating systems by accessing the keyboard’s simple text files configuration files.

These instructions apply to the base Advantage360 model feature the SmartSet Programming Engine. If you have the Professional model with the ZMK engine stop reading and visit kinesis.com/support/kb360pro.

2.0 Direct Programming Overview

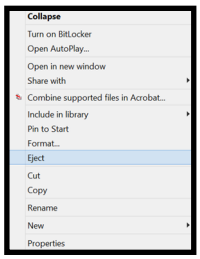
The Advantage360 has 9 customizable Profiles which comprise 9 sets of layouts and indicator lighting configurations. Each of these configurations are stored in a set of folders on the keyboard (the “v-Drive”) as a series of simple text files (.txt). During onboard programming the keyboard automatically reads/writes to these files “behind the scenes”. The unique thing about the 360 is that power users can “connect” (aka “mount”) the v-Drive to their PC and then directly edit these configuration files in Windows, Linux, Mac, and Chrome.

Each time a remap or macro is created in a Profile, it is written to the corresponding layout.txt file as a discrete line of “code”. And the function and color of each of the 6 RGB LEDs is controlled in the corresponding led.txt file.

Note: Do not edit the settings.txt file.

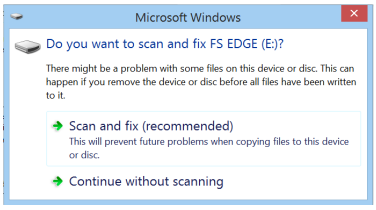
3.0 Before you Begin

3.1 Power Users ONLY



Direct editing requires learning to read and write a custom syntax. The insertion of incorrect characters into any of the configuration files can have unintended consequences and could cause temporary problems with even basic keyboard operation. Read the Quick Start Guide and User Manual first and proceed with caution.

3.2 Always Eject the v-Drive before disconnecting the v-Drive



The v-Drive is just like any other flash drive you connect to your PC. If you remove it suddenly while the PC is still accessing the drive contents you can cause file damage and corrupt your keyboard. To protect the v-Drive, always save and close all configuration files, and then use the appropriate eject protocol for your operating system *before* “disconnecting” the v-Drive with the onboard shortcut. If your PC refuses to eject the drive, ensure all files and folders are closed and try again.

Windows Eject: Save and close any .txt files you have been editing. From File Explorer, navigate back to the top level of “ADV360” removable drive and right click the drive name and then select Eject. Once you receive the “Safe to Eject” notification you may proceed to closing the v-Drive with the onboard shortcut. Failure to eject can result in a minor drive error that Windows will ask you to repair. The “Scan and Repair” process (shown at right) is quick and easy.

Mac Eject: MacOS does not support ejecting the v-drive so users can safely ignore the drive eject error.

3.3 Non-US Users

Your computer must be configured for the English (US) keyboard layout. Other language drivers use different codes/positions for certain keys which are critical for programming characters such as [], {} and >.

3.4 Simple Text Files ONLY

Do not save configuration files in the Rich Text Format (.rtf) as special characters can cause syntax errors.

3.5 Firmware update may be required

Some of the features described in this guide may require a firmware update. Download firmware and get installation instructions here: kinesis.com/support/kb360/#firmware-updates

3.6 Mac Users

Note that due to the way newer versions of macOS handle removable storage drives, “junk” files may be created after editing configuration files. Those junk files can be

4.0 Direct Programming Layouts

The 360 features 9 configurable Profiles, each with its own corresponding “layout” (1-9). The nine default layouts are saved as separate .txt files in the “layouts” subfolder on the v-Drive. Only custom remaps and macros are saved to the file, so if no changes have been made to a layout, the file will be empty and the keyboard performs “default” actions. Users can either write code from scratch or edit existing code using the syntax rules described below. *Note: Deleting a layout file will permanently delete its stored remaps & macros, but the keyboard will automatically regenerate a blank layout file.*

Note: Profile 0 is non-programmable and thus does not have a corresponding layout.txt file.

4.1 File Naming Convention

Only the nine numbered layouts can be loaded to the Advantage360. Additional “backup” layouts can be saved as .txt files with descriptive names, but they cannot be loaded to the keyboard without renaming them first.

4.2 Syntax Overview– Position & Action Tokens

Remaps and macros are encoded in a layout file using a proprietary syntax. Each of the keys on the keyboard (other than the SmartSet Key) has been assigned a unique “Position” token used to identify that key for programming in either layer (see Position Token Map in Appendix A).

Each keyboard & mouse action supported by the 360 has been assigned a unique “Action” token corresponding to a standard USB “scan code”.

View supported actions and tokens here: kinesis.com/support/kb360/#manuals

To successfully re-program a key, the user must use the syntax to designate the physical key (via a Position Token) and assign one or more key actions (via Action Tokens). The “>“ symbol is used to separate Position Tokens from Actions Tokens. Each individual token is surrounded by brackets. Examples:

- Remaps are encoded with Square Brackets: [position]>[action]
- Macros are encoded with Curly Brackets: {trigger key position}{modifier co-trigger}>{action1}{action2}…

Write your remap under the desired “Layer Header” to assign it to that layer:

<base>

<keypad>

<function1>

<function2>

<function3>

4.3 Layout Programming Tips

- If the keyboard cannot understand the desired remap, then the default action will remain in effect
- Do not mix and match square and curly brackets in a single line of code
- Separate each line of code with Enter/Return, but note that additional line-breaks can cause issues
- The order in which the lines of code appears in the .txt file does not generally matter, except in the event of conflicting commands, in which case the command closest to the bottom of the file will be implemented.
- Tokens are not case-sensitive. Capitalizing a token will not produce the "shifted" action.
- A line of code can be temporarily disabled by placing an asterisk (*) at the beginning of the line.

4.4 Position Tokens

Generally speaking, position tokens are defined by the basic QWERTY Windows action for the key in the default layout. In some cases tokens have been modified for clarity and/or ease of programming.

- *Example: The Hotkey 1 position is:* *[hk1]>…*

4.5 Programming Remaps

To program a remap, encode the position token and one action token in square brackets, separated by ">".

Remap Examples:

1. *Hotkey 1 performs Q:* *[hk1]>[q]*
2. *Escape key performs Caps Lock:* *[esc]>[caps]*

Shifted Actions: Shifted characters (e.g., "!") cannot be produced by a Remap. To produce a shifted key action, it is necessary to encode it as macro which includes both the down and up stroke of the shift key surrounding the basic key action. Downstrokes are indicated by placing a "-" inside the bracket and upstrokes are indicated by placing "+". See example macro 1 below.

4.6 Programming Macros

To program a macro, encode the "trigger keys" to the left of the ">" in curly brackets. Then encode one or more Action Tokens to the right of the ">" in curly brackets. Each macro can include approximately 300 Action tokens and each layout can store up to 7,200 total macro tokens spread across up to 100 macros.

Trigger Keys: Any non-modifier key can be trigger a macro. A co-trigger can be added by encoding a modifier to the left of ">". See example 1 below. *Note: Windows co-triggers are not recommended.* Write your macro under the desired "Layer Header".

Individual Playback Speed Prefix {s_}: By default, all macros play at the selected default playback speed. To assign a custom speed for improved playback performance for a given macro you can use the "Individual Playback Speed" prefix "{s_}". Choose a number from 1-9 corresponding to the speed scale shown Section 4.6. The speed prefix should be placed to the right of the ">" before the macro content. See example 2 below.

Multiplay Prefix {x_}: By default, all macros playback continuously while the trigger key is held. To override the repeat feature and restrict a macro to playback a specific number of times you can use the "Macro Multiplay" prefix "{x_}". Choose a number from 1-9 corresponding to the number of times you want the macro to replay. The multiplay prefix should be placed to the right of the ">" before the macro content. See example 3 below. If a macro is not playing back properly, try assigning a Multiplay value of 1. The macro may actually be firing multiple times before you are releasing the trigger key. See example 3 below

Timing Delays: Delays can be inserted into a macro to improve playback performance or to produce a mouse double-click. Delays are available in any interval between 1 and 999 millisecond ({d001} & {d999}), including random delays ({dran}). Delay tokens can be combined to produce delays of various durations.

Macro Examples:

1. Tab key performs “Hi” with a capital H: {tab}>{-lshft}{h}{+lshft}{i}
2. Hotkey 4 + Left Ctrl performs “qwerty” at speed 9: {lctrl}{hk4}>{s9}{q}{w}{e}{r}{t}{y}
3. Hotkey 1 increases the volume 3 notches: {hk1}>{x3}{vol+)

4.7 Tap and Hold Actions

With Tap and Hold, you can assign two unique actions to a single key based on the duration of the keypress. Designate the Position Token in the appropriate layer, then the Tap action, then the timing delay from 1 to 999 milliseconds using the special Tap and Hold token ({t&hxxx}), then the Hold Action. Due to inherent timing delays, Tap-and-Hold is not recommended for use with alphanumeric typing keys. Not all key actions support Tap-and-Hold.

Note: For most applications, we recommend a timing delay of 250ms.

Tap and Hold Example:

- *Caps performs Caps when tapped and Esc when held longer than 500ms: [caps]>[caps][t&h500][esc]*

5.0 Direct Programming RGB LEDs

The 360 features 3 programmable RGB LEDs on each key module. The nine default lighting effects are saved as separate .txt files in the “lighting” subfolder on the v-Drive. The default assignments are shown below. *Note: If the file is blank, the indicators will be disabled.*

Left Key Module

Left = Caps Lock (On/Off)

Middle = Profile (0-9)

Right = Layer (Base, Kp, Fn1, Fn2, Fn3)

Right Key Module

Left = Num Lock (On/Off)

Middle = Scroll Lock (On/Off)

Right = Layer (Base, Kp, Fn1, Fn2, Fn3)

5.1 Define your Indicator

The 6 indicators are defined with a basic position token

- Left Module Left LED: [IND1]
- Left Module Middle LED: [IND2]
- Left Module Right LED: [IND3]
- Right Module Left LED: [IND4]
- Right Module Middle LED: [IND5]
- Right Module Right LED: [IND6]

5.2 Define your Function

A variety of functions are supported and more may be added in the future.

- Disable LED: [null]

- Active Profile: [prof]
- Caps Lock (On/Off): [caps]
- Num Lock (On/Off): [nmlk]
- Scroll Lock (On/Off): [sclk]
- Active Layer:
    - Base: [layd]
    - Keypad: [layk]
    - Fn: [lay1]
    - Fn2: [lay2]
    - Fn3: [lay]

5.3 Define your Color(s)

With the exception of Layer, each function can be assigned a single color value using a 9 digit value corresponding the R-G-B value of the desired color (0-255). The Layer function supports the assignment of up to 5 colors, one for each layer.

5.4 Syntax

Each indicator is encoded in much the same way of a basic remap. Use the indicator position token, the “>“ and then the function, and then the color. For the Layer LED you will need to write a separate line of syntax for each layer

[IND_]>[FUNC][RRR][GGG][BBB]